Discovery Engine for Finding Hidden Connections in Prose Comprehension from References

Amal Babour¹, Javed I. Khan², Fatema Nafa³, Kawther Saeedi⁴, Dimah Alahmadi⁵
Faculty of Computing and Information Technology
Department of Information Systems, King Abdulaziz University, Jeddah 21589, Saudi Arabia¹, ⁴, ⁵
Department of Computer Science, Kent State University, Kent, OH 44240 USA²
Department of Computer Science, Salem State University, Salem, MA 01970 USA³

Abstract—Reading is one of the essential practices of modern human learning. Comprehending prose text simply from the available text is particularly challenging as in general the comprehension of prose requires the use of external knowledge or references. Although the processes of reading comprehension have been widely studied in the field of psychology, no algorithm level models for comprehension have yet to be developed. This paper has proposed a comprehension engine consisting of knowledge induction which connects the knowledge space by augmenting associations within it. The connections are achieved through the automatic incremental reading of external references and the capturing of high familiarity knowledge associations between prose concepts. The Ontology Engine is used to find lexical knowledge associations amongst concept pairs, with the objective being to obtain a knowledge space graph with a single giant component to establish a base model for prose comprehension. The comprehension engine is evaluated through experiments with various selected prose texts. Akin to human readers, it could mine reference texts from modern knowledge corpuses such as Wikipedia and WordNet. The results demonstrate the potential efficiency of using the comprehension engine that enhances the quality of reading comprehension in addition to reducing reading time. This comprehension engine is considered the first algorithm level model for comprehension compared with existing works.

Keywords—Knowledge graph; ontology engine; text comprehension; text summarization; Wikipedia; WordNet

I. INTRODUCTION

Text comprehension is a form of knowledge acquisition whereby readers interact with text and relate the ideas represented to their knowledge and experiences [1]. Generally, reading a single text does not qualify readers to achieve the required level of comprehension. This is because the comprehension process depends largely on reader knowledge or additional knowledge acquired from external sources. A well written text embeds a set of cues to build up a coherent representation of the text. Although the text normally presents a set of related concepts, this does not qualify readers to achieve complete comprehension. In this paper, the focus is on prose comprehension. Prose is a type of text which includes complex concepts manifesting particular meanings of a specific domain with associations among concepts. As prose comprehension is difficult to achieve from a simple reading of the text alone, external knowledge is required to understand the text. The external knowledge readers need is known as prior knowledge [2]. This prior knowledge is different from one to one comprehension. Sometimes, readers may not even have a minimum level of prior knowledge about a specific topic, making it necessary for them to obtain it by using external sources. Lexicons and external references are two common examples of acquiring additional knowledge. By drawing upon prior knowledge, readers fill the knowledge gap in the prose by connecting the prose contents with their prior knowledge. The process of prose comprehension is flexible to implement and incorporates external information sources into the prose text. An example of such a process is integrating an encyclopedia with linguistic information/dictionaries.

Although human readers tend to consult external knowledge mediums such as books, Wikipedia, or journal articles to bridge this knowledge gap, such a process is often time consuming and tedious. It requires reading a large amount of text and then finding the relevant portions of the reference to catalyze understanding. While the mental process behind such knowledge induction is intriguing, adopting computational algorithms can help more effective reading comprehension of prose by automatically capturing relevant text pieces from external references and relevant knowledge association of the prose concepts as a summary. This can leverage greater overall comprehension process of a prose text.

The topic of the human reading rate has been widely studied. Carver’s reading model identifies the relationship model between reading and comprehension [3]. He defines five reading processes: memorizing, learning, rauding, skimming, and scanning. The reading rate of an individual varies on the basis of material difficulty and reading objective [3, 4]. For example, skimming may be used when a reader requires an overview of the material, whereas learning may be used when a reader requires comprehension of the material. Carver’s reading model provides a reading rate that measures how many words are read per minute (wpm) for each of the five processes. For memorization, the rate is 138 wpm, whereas for learning it is 200 wpm, and for rauding the rate is 300 wpm. For skimming, the rate is 450 wpm, and finally for scanning it is 600 wpm. For example, if readers read a text of 2000 words for the purpose of learning, it takes about 10 minutes for them to read and comprehend it.

The previous work focused on generating summarized texts and/or recruiting participants to analyze their level of
comprehension. To the best of knowledge, no existing work proposing an algorithm level model for comprehension. Accordingly, the contribution of this paper can be summarized as follows:

- Develop an algorithm level model comprehension engine to enhance prose comprehension.
- Evaluate the proposed model through set of graph metrics.

The proposed comprehension engine contributes to the reading rate that readers may need to acquire specific knowledge from reference texts. This engine is based on the Knowledge Induction Process which targets increasing knowledge comprehension through two steps: 1) the incremental reading of external reference texts and giving an extractive summary of each by capturing of the highest familiarity knowledge associations amongst prose concepts. One of the distinct features of the proposed algorithm is that it captures the highest familiarity knowledge along with the fewest associations pertaining to prose concepts through a minimum number of external concepts; and 2) using Ontology Engine to find lexical knowledge associations amongst concepts. This can save readers time and increase their efficiency, which are two main advantages of the comprehension engine.

The rest of the paper is structured as follows: Section II provides an overview of the related work. Section III introduces the development of the comprehension engine. Section IV presents an evaluation of the proposed engine. Section V presents the materials used in the experiment and the results, and the final section (Section VI) discusses the conclusion and the future work.

II. RELATED WORK

Considerable research has been done in text summarization and text comprehension in recent years. Some of the research introduced valuable techniques used to produce extractive and abstractive summarization. Extractive summarization summarizes the text by extracting sentences containing salient information from the text itself, while abstractive summarization summarizes the text by paraphrasing the text using words that might not in the text [5]. In recent years, various approaches have been developed for automatic summarization and have been applied widely in different domains.

Van Lierde and Chow [6] combined fuzzy and statistics approaches to obtain extractive summarization. The fuzzy based technique contained manually generated rules where the rules were proceeded based on the length of the sentence. By using these rules, all the sentences were assigned with a weight value ranging from zero to one, where the weight for each sentence was used as a feature in the fuzzy inference system. The authors used such a system to perform the summarization through a number of fuzzy-logic-based analyzers. The work had an advantage of taking into consideration the linguistic variables and human perception.

Hernández-Castañeda et al. [7] proposed a method for extractive automatic text summarization (EATS). The method based on the conversion of the text into numerical vectors by applying different generation methods. The vectors are grouped into clusters based on measuring proximity among the vectors. Latent Dirichlet allocation (LDA) was used to obtain the key sentences in each cluster that make up the summarized text.

Furthermore, Azadani et al. [8] used graph-based summarization techniques to produce extractive summarization. The technique represented a document as a graph in which a node can correspond to various semantic units including words, phrases, concepts or sentences, whereas an edge demonstrated the relation in connectivity between the nodes. The authors used the frequent itemset mining algorithm to extract the summary.

For summarizing multiple documents, Fuad et al. [9] used two techniques, sentence clustering and neural sentence to produce abstractive summarization. In sentence clustering techniques, they used a deep neural network architecture to represent text. In neural sentence, they applied seq2seq encoder-decoder technique. The proposed method took a related ordered set of sentences and produced a single sentence by merging the input sentence from multiple documents; the output was only one sentence as a summary.

For the purpose of text summarization and comprehension, Ding et al. [10] proposed a text summarization generation model to enrich representing the information of the original text and improve the text comprehension. Their method was based on seq2seq through a dual-encoder, the Gain-Benefit gate for decoding, and the probability distributions of the keywords in the text.

In [11], Caglierio et al. proposed a method named TESTdriven SUMmarization (TestSum) to recommend text summarization based on a learner’s level of comprehension. The method provides multiple-choice tests to assess the learner’s comprehension level of different topics. It performs a Multilingual Weighted Itemset-based Summarizer (MWI1Sum) that relies on frequent itemset mining sentence selection and ranking to generate the material summary. It recommends a personalized summary to learners who did not pass the multiple-choice test.

All the previous text summarization research focused on reducing the volume of the text by capturing or rephrasing the most important sentences that include main keywords and present it to the reader. On the other hand, the current text comprehension research based on recruiting participants and applying descriptive statistics to analyze their performance. Thus, this paper has introduced a text summarization method that may increase the number of concepts in the summary in some cases for the purpose of text comprehension.

III. KNOWLEDGE INDUCTION PROCESS

“Knowledge graph is an emerging technology for massive knowledge management and intelligent services in the big data era” [12]. It provides an ideal technical solution to realize the integration of knowledge sources by incorporating noisy information and connecting the fragmented pieces of knowledge from multiple sources in a consistent way. In this paper, a knowledge graph called the Illuminated Knowledge
Graph (IKG) is proposed to illuminate the relationships among prose concepts using multiple knowledge sources for the purpose of comprehension. These ideas and goals were the inspiration for the graph name.

The IKG is a graph which captures the state of learning process. It shows prose concepts (CL) and associations amongst the concepts. The associations can be found by reading the prose (LTX) and reading some parts relevant to the concepts as a summary from external reference texts (RTX) as well as an Ontology Engine (OE). Beside the CL concepts, the graph could include external concepts belong to RTX or OE in the associations added from them to connect the prose CL concepts. A directed graph presents IKG = (C, E), where C is a concept set and E is an edge set. A concept \( c_j \) includes set of senses \( s_{i,j} \), where \( i \) is the number of concept and \( x \) is the number of senses. An edge connects two concepts through a specific sense of each concept. The edge represents a sentential relation between the concepts. It can be a syntactical explicit and/or an Ontology Engine relation, where the latter is one of the six types of word relations: Hyponym, Hypernym, Holonym, Instance, Meronym, or Synonym. Fig. 1(a) illustrates an example of five concepts in IKG. \( c_1 \)={\emph{Country}}, \( c_2 \)={\emph{State}}, \( c_3 \)={\emph{freedom}}, \( c_4 \)={\emph{Ohio}} and \( c_5 \)={\emph{Liberty}}, where \( c_1 \), \( c_2 \) and \( c_3 \) are the CL belonging to LTX. \( c_2 \) belongs to a RTX. \( c_3 \) belongs to an OE.

A Knowledge Path (K) represents the relationship path between two concepts. It is represented as a sequence of edges connecting concepts \( c_i \) and \( c_j \) in a preserved sense. The concepts \( c_i \) and \( c_j \) belong to CL. The concepts in the middle may be external to CL. The following are examples of geometric paths:

1) \( \{c_1\rightarrow s_{1,1} ; \text{Synonym: } s_{2,1} \rightarrow c_2 \rightarrow s_{3,1} ; \text{Instance: } s_{4,1} \rightarrow c_4 \}. \)
2) \( \{c_1\rightarrow s_{1,1} ; \text{Synonym: } s_{2,1} \rightarrow c_2 \rightarrow s_{2,2} ; \text{syntactic explicit: } s_{4,1} \rightarrow c_4 \}. \)
3) \( \{c_1\rightarrow s_{1,1} ; \text{Synonym: } s_{2,1} \rightarrow c_2 \rightarrow s_{2,2} ; \text{Hyponym: } s_{3,1} \rightarrow c_3 \rightarrow s_{3,1} ; \text{Hyponym: } s_{5,4} \rightarrow c_5 \}. \)

\( c_i \) refers to the \( i \)th concept, \( s_{i,j} \) is the \( j \)th sense of the \( i \)th concept. For example, \( c_1 \) is the first concept and \( s_{1,1} \) is the first sense of the first concept. Fig. 1(b, c, d, e) represents examples of geometric paths. The knowledge paths can be derived from geometric paths. For example:

1) \( \{c_1\rightarrow s_{1,1} ; \text{Synonym: } s_{2,1} \rightarrow c_2 \rightarrow s_{3,1} ; \text{Instance: } s_{4,1} \rightarrow c_4 \}. \)
2) \( \{c_1\rightarrow s_{1,1} ; \text{Synonym: } s_{2,1} \rightarrow c_2 \rightarrow s_{2,2} ; \text{syntactic explicit: } s_{4,1} \rightarrow c_4 \}. \)
3) \( \{c_2\rightarrow s_{2,2} ; \text{Hyponym: } s_{3,1} \rightarrow c_3 \rightarrow s_{3,1} ; \text{Hyponym: } s_{5,4} \rightarrow c_5 \}. \)

The three paths that mentioned above are considered knowledge paths, as the incoming and the outgoing senses of each concept are preserved. Fig. 1(b, c, e) represents examples of knowledge paths.

A prose can be often rich with concepts using domain specific terms. While reading a prose, non-specialist readers find these terms difficult to comprehend. They may face these difficulties while reading prose in any domain such as science, medical, finance and technology. Therefore, external knowledge helps the comprehension of prose by the integration with the prose concepts [13, 14]. This is not always a straightforward process, as readers’ prior knowledge may vary. Inexperienced readers may get overwhelmed by the amount of diverse external knowledge sources and types available since the latter can range from reference texts, dictionaries, and papers to conversations with experts. Identifying the right source of knowledge from a vast range of information can be a time consuming and exhausting task.

Therefore, this work proposes the Knowledge Induction Process to enhance prose comprehension. The process is designed to capture knowledge missing in a prose. The external knowledge source is captured through augmenting knowledge associations of prose concepts using reference texts and an Ontology Engine. The proposed solution assumes that concepts are known prior to the process of finding the knowledge associations amongst the concepts. The process reads appropriate parts from relevant reference texts and then captures a summary of the highest familiarity knowledge associations connecting the prose concepts. The Ontology Engine captures lexical knowledge associations for every concept pair. This process is formally presented in the following way: Given a prose LTX for comprehension, a set of prose concepts CL. CL= \{c_0, c_1,\ldots, c_n\} \subseteq LTX, a set of reference texts RTX, and an Ontology Engine OE. Find the IKG to represent knowledge associations amongst CL. The IKG is built through two fundamental techniques, namely a generation concepts representation and a reference consultation.

A. Generation of Concept Representation

To understand a prose text, a reader may break it up into a set of concepts and then find knowledge associations amongst these concepts. In order to find new or missing knowledge associations among the prose concepts, this process can be automated by applying a computational representation model. It is assumed that this enhances the prose comprehension. A graph is used to represent the concept and the associations, where each concept is represented by a node and each sentential relation between two concepts is represented by an edge.

A Syntactical Explicit Graph generator function KG() is used to convert a prose text LTX to a knowledge graph G_k, and a reference text RTX to a reference knowledge graph GR. For each sentence in LTX/RTX, the function searches for pairs of concepts \( c_i \) and \( c_j \) if a word or a sequence of words found \( e_b, b=1,2,\ldots,n \) between them in the same sentence, where in LTX, \( c_i \) and \( c_j \) belong to CL and in RTX, \( c_i \) and \( c_j \) belong to the reference text noun concepts. The distance between \( c_i \) and \( c_j \) is <= L, where L is the maximum number of words allowed between \( c_i \) and \( c_j \). If it does, the function saves \( [c_i, e_b, c_j] \) to be an edge in the graph representing a syntactical relation between \( c_i \) and \( c_j \).

To allocate the most familiar knowledge associations that connect the prose concepts, it is crucial to evaluate the familiarity of knowledge associations. These can be calculated through the edge weight in the knowledge graph. Calculating the weight/familiarity value is based on the type of the sentential relation between concepts. The sentential relation types reflect the structures between any pair of concepts. The weight \( w_{ij} \) is calculated by (1) where \( f_{ij} \) is the frequency of the
relation type between concept $c_i$ and $c_j$ attained from the "Gutenberg Project" [15]. The "Gutenberg Project" is an online resource offering over 53,000 free e-books with expired US copyrights. This online archive is a very popular dataset, frequently applied in text mining research [16, 17].

$$w_{i,j} = -1/\left(\frac{1}{\log \left(\frac{f}{10^9}\right)}\right)$$ (1)

The familiarity value of a sequence of words between $c_i$ and $c_j$ is calculated based on the lowest weight of words sequence. The log of a word frequency is divided by $10^9$, as word frequencies are in the millions. The result is multiplied by -1 to avoid negative values. High frequency refers to high familiarity of a relation type. An inverse relationship between $f$ and $w$ indicates that the higher the frequency, the less its weight or the less its cost.

Table I explains five types of the sentential relations between concept pairs. (1) is where a single word $e_b$ appears between $c_i$ and $c_j$, $b=1$. (i.e. Carbon releases carbonization), here $c_i = \text{Carbon-Carbon}$, $c_j = \text{carbonization}$, and $e_b = \text{releases}$. The weight of the edge is identified by considering the frequency within (1) as the frequency of the word $e_b$. (2) is where multiple words $e_b$ appear between $c_i$ and $c_j$, $b>1$ (i.e. Ethane is structurally the simplest hydrocarbon), where $c_i = \text{Ethane}$, $c_j = \text{hydrocarbon}$, $b=2$, $e_b = \text{structurally}$, and $e_b = \text{simplest}$. The weight of the edge is computed by considering the weight of $e_b$ and $e_b$ separately and then allocating the minimum weight. (3) is a class/subclass, wherein the sentential relation is either a hypernym or hyponym (i.e. Ethane is a hyponym of hydrocarbon), $c_i = \text{Ethane}$, $c_j = \text{hydrocarbon}$ and $e = \text{hyponym}$. The weight of the edge is determined by considering the frequency within (1) as the frequency of the word class. (4) is a part or subpart, wherein the sentential relation is either a holonym or meronym (Hydrogen is a holonym of water), $c_i = \text{Hydrogen}$, $c_j = \text{water}$ and $e = \text{holonym}$. The weight of the edge is determined by considering the frequency within (1) as the frequency of the word part. (5) is a synonym, if the sentential relation is a synonym (Ethane is a synonym for $C_2H_6$), $c_i = \text{Ethane}$, $c_j = \text{C}_2\text{H}_6$ and $e = \text{synonym}$. The frequency of the synonym relation is supposed to be 1. Therefore, the weight of the edge is determined by considering 1 in the frequency given by (1).

To generate an IKG, the Knowledge Induction Process performs six steps as presented in Fig. 2:

1) Converting a prose LTX to a prose knowledge graph $G_0$ representing the syntactical relation between each pair in the prose concepts CL by performing a Graph generator function KG().

2) Converting a reference text RTX$_i$ to a reference knowledge graph $G_R_i$ representing the syntactical relation between each pair of the RTX$_i$ concepts by performing the same Graph generator function KG() used in step 1.

3) Extracting the highest familiarity knowledge path(s) of RTX$_i$ connecting CL from GR$_i$ by performing the Terminal to Terminal Traffic Steiner Tree function TTTST(). The extracted path(s) is called a Terminal to Terminal Traffic Steiner tree(s) TTTST and it is represented in a graph called $GU_i$.

4) Joining $G_0$ and $GU_i$ in a graph called $Gtemp$ representing the current state of the assimilated knowledge among CL by performing an assimilation function $Gassmilation()$.

5) Finding OE-Knowledge-Path(s) connecting each pair of $Gtemp$ concepts by performing the OE-Knowledge-Paths function KPOE(). The found paths are represented in a graph called $GW_i$.

6) Joining $Gtemp$ and $GW_i$ in an Illuminated Knowledge Graph IKG$_i$ representing the current state of the assimilated knowledge amongst CL by performing the assimilation function $Gassmilation()$.

Each time the process reads a new RTX$_i$, it performs steps 2 to 6 where, in step 4, $G_0$ is replaced with IKG$_i$. Fig. 3 shows
an example explaining the impact of using reference texts and 
an Ontology Engine for adding knowledge associations 
amongst a set of prose concepts about 'Ethane' chemical 
compound, where CL= \{ethane, hydrocarbon, hydrogen, 
carbon, carbon-carbon and carbonization\}.

<table>
<thead>
<tr>
<th>Sentential relation type</th>
<th>Sentential relation structure</th>
<th>$w_{ij}$ value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntactical Relation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_i$</td>
<td>$e_1$</td>
<td>$c_j$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_i$</td>
<td>$e_1 \ldots e_2$</td>
<td>$c_j$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OE Relation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_i$</td>
<td>hyponym</td>
<td>$c_j$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_i$</td>
<td>holonym</td>
<td>$c_j$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_i$</td>
<td>synonym</td>
<td>$c_j$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2. Comprehension Engine.
Fig. 3. Connecting CL Process using Reference Texts and Ontology Engine. (a) a set of Prose Concepts at the Initial Prose. (b) Knowledge Path K from the Initial Prose LTX. (c, e, f) Additional Knowledge Path K Extracted from RTX1, RTX2, and RTX3. (d) Additional Knowledge Path K Extracted from Ontology Engine OE.
B. Reference Consultation

1) Highest familiarity knowledge extraction using a reference text: Knowledge within a prose is limited, as readers cannot capture the knowledge relevant to all concepts required to comprehend the prose within the prose text only. Therefore, external reference texts are required to capture knowledge association amongst prose concepts. The reference texts are used to fill the knowledge gap of prose.

There are a number of Steiner tree versions that can be useful for identifying knowledge associations amongst the prose concepts [18, 19]. The main role of a Steiner tree in the proposed engine is to capture the highest familiarity knowledge paths amongst the prose concepts CL from each reference text GR, with a minimum cost and a minimum number of exported concepts, where the sentences of the captured knowledge paths are considered the summary of the reference text.

Given a connected, undirected reference knowledge graph GR=(C, E), where C is a set of reference text concepts and E is a set of edges representing the relations amongst the graph concepts, the weight for each edge w_{ij} reflects its cost, where the cost here expresses the familiarity value of the sentential relation between the concepts of the edge ends c_i and c_j as shown in Fig. 4(a). Fig. 4(b) shows the set of prose concepts CL that need to find the highest familiarity knowledge paths amongst them in GR. Minimum Steiner Tree (MST) is an approach based on finding knowledge with the minimum cost amongst prose concepts [18]. The cost of MST is calculated by \( \sum (w_{ij}) \), where i, j \in C and i \neq j. Sometimes, connecting the CL in a MST requires the addition of external concepts to CL. This case can be found in Fig. 4(c), C= \{A, B, C, D, E\}, CL= \{A, C, E, D\}. The returned Steiner tree is \{E, D, B, A, C\} and it costs 10.

Suppose there is traffic (Tr) between c_i and c_j, where the traffic indicates the comprehension amount of the relation between the two concepts. In this case, the aim of the Steiner tree is to reduce the traffic amongst CL, whereby low traffic means high comprehension. Here, the Steiner tree is called Terminal to Terminal Traffic Steiner Tree (TTTST). Its cost is \( \sum (w_{ij} \times T_{ij}) \), where i, j \in C and i \neq j. In Fig. 4(d), supposes that the traffic between each two concepts is \( T_{ij}=1 \) and the traffic between concept E and concept A is \( T_{E,A}=100 \). In this case, the traffic weighted cost for the returned Steiner tree is \( ((4 \times 100) + (1 \times 1) + (3 \times 1) + (2 \times 1)) = 406 \).

Fig. 5 represents the TTTST algorithm in the following way. The algorithm input is GR, and CL. GR is a reference knowledge graph representing the syntactical relation between each pair of RTX, concepts. The algorithm output is GU, which is a tree extracted from GR, representing the highest familiarity knowledge path(s) connecting CL.

```
Def TTTST ( ):

Input: GR, CL

Output: TTTST: Terminal to Terminal Traffic Steiner Tree

1. /Initialization
2. for each co in GR:
3. if CL= null:
4. for each concept c in co
5. prev[c]= -1
6. cost[c]= INFINITY
7. Visited[c]=False
8. Q=null
9. while any concept from CL:
10. enqueue(Q, c)
11. while Q is null:
12. c = dequeue(Q)
13. Visited[c]=True
14. if c in CL:
15. cost[c]= 0
16. add c to M
17. remove c from CL
18. for each neighbour c_i of c:
19. if c_i not in Q and Visited[c_i]=False:
20. enqueue(Q, c_i)
21. temp= cost[c] + w_{c,c_i}
22. if temp <= cost[c_i] and c in CL:
23. prev[c_i]=c // a knowledge path with minimum external concepts to c_i is found
24. cost[c_i]=temp
25. else if temp < cost[c_i]
26. prev[c_i]= // a less costly knowledge path to c_i is found
27. cost[c_i]=temp
28. TTTST = getPaths(M), prev[ ]
29. return TTTST
```

Fig. 5. Terminal to Terminal Traffic Steiner Tree Algorithm.

For each component co in GR, the algorithm uses a queue data structure Q to store each visited concept with its neighbors. It initializes Q by picking any concept from CL as the source s. Then, it initializes the cost between s and each concept c in the co to INFINITY and the previous concept prev of each c to -1. In the loop iteration, it dequeues the first concept c in Q, marks it as visited, and checks if c belongs to CL. If so, it updates its cost to 0, adds it to M where M stores the found CL concepts, and removes it from CL. Then, it
enqueues all the neighbors’ c, s of concept c if they are marked as non-visited, assigns prev and calculates the cost for each of them. If the current c, cost is equal to the previous one and c belongs to CL that means a knowledge path with minimum external concepts is found. Then, c, s prev is updated to the new concept. However, if the current c, cost is less than its previous one that means a less costly knowledge path to c, is found wherein less cost means high familiarity. Then, c, s prev and cost are updated to the new lesser values and the process is repeated until the queue is emptied. If all items in co are checked, getPaths() constructs the TTTST from M and prev. The returned TTTST is represented in GU, [20]. Suppose the graph shown in Fig. 6 is a reference knowledge graph GR, If CL = {c, c, c} and the traffic for each edge Tr = I, then the TTTST returned by algorithm will be {c, c, c, c, c, c, c}.

2) Knowledge extraction using ontology engine: A useful source of knowledge is the Ontology Engine. An Ontology Engine to provide lexical knowledge associations between any two concepts by using OE-Knowledge-Paths function KPOE( ) is utilized. This helps in adding a new type of knowledge which can contribute to increasing knowledge and improving comprehension. The OE-Knowledge-Path is a sequence of edges connecting any two concepts in a preserved sense. Each edge represents an ontological relation between its ends, where the ontological relation for each edge is one of the following relations: (1) synonym; (2) hypernym; (3) hyponym; (4) holonym; and (5) meronym.

Fig. 7 represents the KPOE algorithm. It uses an Ontology Engine to search for OE-Knowledge-Path(s) connecting concept s and concept t, where s and t are the first and the last concepts in the path and the length of the path is less than or equal to a. The input of the algorithm is s, t, R, α, and relationalGraph. R is a dictionary that holds all the ontological engine relations in the Ontology Engine, and relationalGraph is a dictionary of all concepts that have any of the ontological relations in R with the last node of the current path. The algorithm does not return the shortest path between s and t because it could be a path of multiple sense concepts. Rather, it searches if the neighbors of s have the same sense of s and have any of the ontological relation in R with s. If so, it searches the neighbors of the neighbors until it reaches t.

NodeQ and PathQ are two queue data structures used in the algorithm. NodeQ keeps the current path that has a concept need to explore its neighbor. PathQ holds the currently created paths. The algorithm starts with s as the current path. The while loop iterates through PathQ paths searching for an OE-Knowledge-Path that connects s and t. In each iteration, it dequeues the first path in PathQ and signs it in NodeQ. Then, it checks whether the last concept in NodeQ matches t. If so, it saves the path in Kpaths as an OE-Knowledge-Path between s and t; otherwise, it checks if the NodeQ length is less than a. If the last concept in NodeQ matches t, for the sense of the NodeQ’s last concept, the function gets all of the concepts that have one of the ontological relations in R with the last concept in NodeQ, and adds them to relationalGraph. A number of paths are created between each relationalGraph concept and the current path. PathQ saves the newly created paths. If all the paths in PathQ are checked and Kpaths do not exist, the function returns ‘Not found’ [20].

Fig. 6. Terminal to Terminal Traffic Steiner Tree Example.

Fig. 7. Discovering OE-Knowledge-Paths Algorithm.
For example, consider $s=$ “country”, $t=$ “group”, and $a=$ 4. The returned OE-Knowledge-Path by the algorithm is country (hyponym) people (hyponym) group. The process of discovering the OE-Knowledge-Path between “country” and “group” is shown in Fig. 8.

IV. COMPREHENSIVE MODEL EVALUATION

An evaluation model is needed to assess the comprehension engine. The statistical characteristics of the knowledge obtained by the comprehension engine need to be measured. However, the process of measuring knowledge is still a difficult area that needs to be explored. “The fluid and intangible nature of knowledge makes its measurement an enormously complex and daunting task” [21].

The paper has proposed the use of graph theoretical metrics that provide a natural way to describe the structure and interplay of graph edges. A set of graph metrics presented in this section are divided into two types: quantitative assessment and organizational assessment. The mentioned types are important for the following reasons. (1) Quantitative assessment calculates rare knowledge growth, knowledge overload, and entropy to elaborate different aspects about the amount of knowledge that can be gained from the graph. (2) Organizational assessment calculates the size of the giant component, cluster coefficient and graph density to study the strength of the connections between concepts and their neighbors and the connectedness of the graph where high connectivity means high comprehension. The following section explains the real comprehension model based on all the mentioned graph metrics.

A. Content of Information

Comprehension enhancement is influenced by the size of obtained knowledge in the IKG. The obtained knowledge graph size can be computed by counting the entire number of concepts $C$ and the entire number of relations $E$ in IKG. Based on the Knowledge Induction Process, the graph starts with the prose knowledge graph $G_0$. It is transformed into IKG$_1$ after reading RTX$_1$, and then into IKG$_2$ after reading RTX$_2$ until it reaches IKG$_{final}$ after reading all the RTX$_i$. At each state, the size of IKG may be increased and knowledge may be augmented. The knowledge growth rate $\lambda$ is computed by (2) where $|IKG|_i$ is the IKG size after reading RTX$_i$, and $|G_0|$ is the prose knowledge graph size.

$$\lambda = \frac{|IKG|_i}{|G_0|}$$ (2)

The knowledge overload rate $\gamma$ may also be increased, and it can be determined by (3):

$$\gamma = \frac{|IKG|_i - |G_0|}{|G_0|}$$ (3)

The amount of information that can be obtained from the knowledge graph can also be calculated by measuring the Entropy $\delta$ of the knowledge graph, where high entropy indicates that the knowledge graph has a high amount of information and vice versa. Based on [22], $\delta$ can be computed by (4):

$$\delta = -\sum_{i=0}^{n} p_i \log(p_i)$$ (4)

Where $p_i$ is the degree distribution probability of concept $c_i$ in the knowledge graph that can be determined by (5):

$$p_i = \frac{d_i}{2|E|}$$ (5)

Where, $d_i$ is the degree of concept $c_i$ and $|E|$ is the entire number of the relations in the knowledge graph. $\delta$ is divided by $\log(n)$ to obtain a measure between 0 and 1, where $n$ is the entire number of the knowledge graph concepts.

B. Knowledge Organization

It is obvious that the existence of relationships amongst concepts in the prose influences comprehension. The greater the increase in the relations among the concepts, the higher the understanding of the relations between them. This falls under a graph organization notion that can be calculated by the size of the giant component [23] that can be calculated using (6):

$$GC = \frac{|C'|}{|C|}$$ (6)

Where $|C'|$ is the number of connected concepts forming a giant component and $|C|$ is the total number of concepts in the knowledge graph. The relation amongst the concepts and their neighbors in the knowledge graph can be calculated by the cluster coefficient $\beta$ [24]. Based on [25], $\beta$ is obtained by using (7):

$$\beta = \sum_{i=0}^{n} \frac{2NIC_i}{d_i(d_i-1)}$$ (7)

Where $NIC_i$ is the neighbors’ interconnection coefficient of concept $c_i$, which denotes to the number of the sentential relations between the first neighbors of concept $c_i$, and $d_i$ is the sentential relations of concept $c_i$ which counts the first neighbors of concept $c_i$. The graph is considered highly clustered when the value is towards 1. The knowledge graph density $\rho$ is another way to study the Knowledge graph organization to explore the completion and the integrations amongst concepts [26]. A complete knowledge graph contains all possible sentential relations $E$ and density equals 1. The graph density $\rho$ is calculated by (8) [27].

$$\rho = \frac{|E|}{|C|(|C|-1)}$$ (8)
Calculating the knowledge growth rate $\lambda$, the knowledge overload $\gamma$ rate, and the cluster coefficient $\beta$ require a reading of the external consultation to perform their calculations. Thus, their values are zero before reading the external references. The following example explains the influence of reading three reference texts with an Ontology Engine for adding knowledge associations amongst the prose concepts. The comprehension model evaluation for the proposed metrics before using external consultations in Fig. 3(b), and after using them in Fig. 3(f), respectively, is shown in Table II.

### Table II. Example of the Comprehension Engine Evaluation Before and After Using Reference External Consultation

<table>
<thead>
<tr>
<th></th>
<th>Before using external consultations</th>
<th>After using external consultations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Growth $\lambda$</td>
<td>0</td>
<td>2.5</td>
</tr>
<tr>
<td>Overlap $\gamma$</td>
<td>0</td>
<td>1.5</td>
</tr>
<tr>
<td>Entropy $\delta$</td>
<td>1.5</td>
<td>0.96</td>
</tr>
<tr>
<td>Giant Component Size GC</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>Cluster Coefficient $\beta$</td>
<td>0</td>
<td>0.92</td>
</tr>
<tr>
<td>Density $\rho$</td>
<td>0.07</td>
<td>0.21</td>
</tr>
</tbody>
</table>

### V. Experiment

#### A. Content Material

An experiment was conducted on three prose texts LTX and eight concepts were selected arbitrarily as the list of the prose concepts CL from each LTX. Wordnet [28] is a reliable Ontology Engine that has been used by many researchers in this area. It is a vast lexical database introduced by George Miller at the Cognitive Science Laboratory at Princeton University that is used as a dictionary of word senses and semantic relations between words [29-31]. This experiment was based on Wordnet Version 1.7 as a general reference text.

The used benchmark reference text articles for this paper were derived from https://en.wikipedia.org. Wikipedia articles were employed as RTX. For each LTX, a number of articles are taken from Wikipedia to represent each concept in CL. For the article allocation, the method presented in [32] to automatically select Wikipedia articles was chosen. Table III displays the selected LTXs as well as the CL for each prose.

For each LTX, the sentential relation among the concepts in its CL is represented in the prose knowledge graph $G_0$. Through the Knowledge Induction Process, the system builds a set of Illuminated Knowledge Graphs $IKG_i$ by scanning all of the RTX, where $IKG_i$ is the sentential relation amongst the concepts in CL after reading a new RTX. The complete list of the used reference text articles can be found in Appendix Table V. All the references texts were accessed on July 24, 2019.

#### B. Results

This section presents the analysis of information gained from the Illuminated Knowledge Graph $IKG_i$ of prose LTX in the Knowledge Induction Process.

The average time needed to read the prose, the eight references attached to the prose and identification of the highest familiarity of sentential relations connecting the prose concepts in each reference are as the following: 4 minutes and 16 seconds for the 1st prose (LTX1), 5 minutes and 18 seconds for the 2nd prose (LTX2), and 2 minutes and 7 seconds for the 3rd prose (LTX3). As noticeable, the machine spends a few minutes to read each prose with its related references.

Fig. 9 displays the information growth $\lambda$ per knowledge graph in each prose LTX. The x-axis represents the prose knowledge graph $G_0$ and the Illuminated Knowledge Graph $IKG_i$ after reading each reference text RTX, whereas the y-axis represents the growth rate $\lambda$. In LTX1, there is a gradual increase in the information after reading reference texts RTX1 to RTX6, even though no new information was inserted after reading RTX7. Interestingly, it started to increase again after reading RTX8. In LTX2, the information kept growing gradually from reading RTX1 to RTX8. In LTX3, information increases varied. The information grew after reading RTX1, while no new information was added after reading RTX2 to RTX8. It increased again after reading RTX9 and RTX10, although no new information was inserted after reading RTX7 and RTX8. This leads to the conclusion that reading references in some cases affect the increase of the knowledge positively, while in other cases these yield no effect.

The information overload rate $\gamma$ in the knowledge graph is shown in Fig. 10. Here, the x-axis represents the prose knowledge graph $G_0$ and the Illuminated Knowledge Graph $IKG_i$ after reading each reference text RTX, and the y-axis represents the overload rate $\gamma$. Similarly, it is clear that information overload varies from being slight to high in the three prose LTX after reading new reference texts RTX.
Fig. 11 represents the entropy $\delta$ per each knowledge graph. The x-axis locates for both the prose knowledge graph $G_0$ and the Illuminated Knowledge Graph $IKG_i$ after reading each reference text $RTX_i$, and the y-axis is the entropy $\delta$. It was observed that $\delta$ in the three proses LTX started with low values, and then it bounced to high values after reading the 1st reference text $RTX_1$. This implies that $RTX_1$ in the three proses LTX contributed to adding a high amount of information. Then, in LTX, the entropy value increased a little after having read $RTX_2$ to $RTX_6$ and $RTX_8$. This means that texts $RTX_2$ to $RTX_6$ and $RTX_8$ inserted a low amount of information, while no new information was inserted when reading $RTX_7$. In LTX, there was a slight increase in entropy after reading texts $RTX_2$ to $RTX_6$; thus, a limited amount of information was inserted. In LTX, there was also a slight increase in the entropy after reading $RTX_2$ and $RTX_8$; a low amount of information was inserted here, and no new information was added after having read texts $RTX_2$ to $RTX_8$. This means that some of the reference texts are highly effective in adding information, while other texts have little effect.

Fig. 12 represents the amount of information that was gained due to contributions of the prose LTX, the reference text RTX, and the Ontology Engine OE concepts in the three proses separately. As shown in Fig. 12(a), for LTX, the entropy value of OE concepts bounced to a greater value than that of the LTX and RTX concepts after reading texts $RTX_1$ to $RTX_8$. This finding suggests that the amount of information inserted by the contribution of OE concepts was higher than the amount inserted by the contribution of the LTX and RTX concepts. In Fig. 12(b), for LTX, the highest entropy value of the LTX, OE, and RTX concepts varied after reading $RTX_1$ to $RTX_8$; this indicated that the highest amount of information obtained by the contribution of the LTX, OE, or RTX concepts was not concentrated on the contribution of any one of them. In Fig. 12(c), for LTX, the entropy value of the OE concepts was the highest amongst the LTX and RTX concepts after having read $RTX_1$ to $RTX_8$. This means that the amount of information inserted by the contribution of the OE concepts is higher than the amount inserted by the contribution of the LTX and RTX concepts. This explains the importance of the LTX, RTX, and OE concepts when inserting information, showing that none of these concepts show greater gains than any other of these concepts.
component GC of the knowledge graph. When the process reads a new RTX, the size of the giant component increases. It is observable that for LTX1, the giant component size moved from 0.25 to 0.85 after reading RTX1, and growing to as high as 1 after reading RTX8. For LTX2, the giant component size was 0.25 in the prose knowledge graph G0, and after reading RTX1, it reached to 1 and the concepts were fully connected. Meanwhile in LTX3, the giant component size bounced from 0 to 0.85 after reading RTX1 and arrived at 1 after reading RTX8. It means that there was an enhancement of prose comprehension by reading reference texts for connecting the prose concepts in a giant component and illuminating sentential relations amongst them.

The breakdown of the total number of concepts C and the number of the sentential relations E in the prose knowledge graph G in the prose knowledge graph G, and the Illuminated Knowledge Graph IKGfinal for each prose LTX, are shown in Table IV, wherein the concepts are from the prose LTX, the reference text RTX, and/or the Ontology Engine OE. As can be seen, there is a great variance in the number of concepts and the number of sentential relations between G0 and IKGfinal. Increasing the size may affect the information size positively. Therefore, increasing the IKGfinal size is a significant sign of the overflowing information in the IKGfinal, which can further support reinforcement of the prose comprehension.

Moreover, Fig. 14 illustrates the clustering coefficient β obtained in each knowledge graph. The x-axis refers to the prose knowledge graph G0 and the Illuminated Knowledge Graph IKG, after reading each reference text RTX, whereas the y-axis represents the clustering coefficient β. It is clear that some of the graphs are highly clustered; this is due to the fact that many of the concepts within these graphs are highly related to one another. For LTX1, the graph tended to be highly clustered, especially after reading RTX1, RTX4, RTX5 and RTX8. In LTX2, the cluster coefficient bounced to a higher value after reading RTX3, lowered slightly after reading RTX4, then slightly increased after reading RTX4 to RTX8. In LTX3, the cluster coefficient, which was higher when reading RTX1 became stable after reading RTX2 to RTX8. It increased again after reading RTX8 and then returned to being steady after reading RTX7 and RTX8. This means that there is enhancement of comprehension when some of the reference texts increase clustering of the concepts. In cases where the texts exercise a limited effect on clustering of the concepts, enhancement of comprehension is not observed.

Fig. 15 gives us a view of the integration amongst the concepts included within each knowledge graph. The x-axis is the prose knowledge graph G0 and the Illuminated Knowledge Graph IKG, after reading each reference text RTX, and the y-axis is the graph density ρ. As the graph reflects, in the three prose LTX, ρ showed a variance amongst the reference texts when inserting sentential relations amongst the concepts, bringing them close together.

Table IV. Breakdown of the Total Number of Sentential Relation and Concepts in the Three Proses

<table>
<thead>
<tr>
<th>1st prose LTX1</th>
<th>2nd prose LTX2</th>
<th>3rd prose LTX3</th>
</tr>
</thead>
<tbody>
<tr>
<td>G0</td>
<td>IKGfinal</td>
<td>G0</td>
</tr>
<tr>
<td>Sentential relation</td>
<td>1 168</td>
<td>1 90</td>
</tr>
<tr>
<td>Number of prose LTX concepts</td>
<td>8 8</td>
<td>8 8</td>
</tr>
<tr>
<td>Number of reference concepts</td>
<td>0 12</td>
<td>0 16</td>
</tr>
<tr>
<td>Number of Ontology Engine OE concepts</td>
<td>0 63</td>
<td>0 21</td>
</tr>
</tbody>
</table>
VI. CONCLUSION

The objective of this paper was to enhance prose comprehension by reducing the time expended on understanding the text and increasing the quality of comprehension. Accordingly, the comprehension engine was developed to read several reference texts, to allocate a summary of the highest familiarity knowledge amongst a set of target concepts in each reference and to provide lexical associations among the concepts by using an Ontology Engine. The comprehension engine was evaluated through set of graph metrics. The output performance of the comprehension engine demonstrates the efficiency of the compression engine in reducing reading time and increasing the quality of comprehension. Furthermore, the proposed engine can have academic implications on students or academic learners by providing them with the most familiar and lexical knowledge that can help in prose comprehension in a short time. Although the proposed engine may help readers to increase their knowledge and improve prose comprehension, at times this can cause mess and give rise to ‘too much knowledge’ in prose comprehension, thus challenging preservation and retention by human memory. Future work will be carried out to develop a second phase of the comprehension engine that efficiently select the less difficult to understand knowledge from the augmented knowledge and present it to readers as an enhanced text. In addition, the impact of using the comprehension engine in improving the comprehension through evaluating the comprehension rate on participants and make a comparison between the two-phase results will be studied.

REFERENCES


APPENDIX

**LIST OF THE REFERENCE TEXT ARTICLES USED IN THE EXPERIMENT.**

<table>
<thead>
<tr>
<th>Title</th>
<th>Link</th>
</tr>
</thead>
</table>